Robust Multi-class Graph Transduction with higher order regularization

International Joint Conference on Neural Network, 2015, Killarney.
http://www.producao.usp.br/handle/BDPI/49417

Robust Multi-class Graph Transduction with Higher Order Regularization

Celso A. R. de Sousa, Gustavo E. A. P. A. Batista
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo, São Carlos, Brazil
email: { sousa, gbatista }@icmc.usp.br

***Abstract*—Graph transduction refers to a family of algorithms that learn from both labeled and unlabeled examples using a weighted graph and scarce label information via regularization or label propagation. A recent empirical study showed that the *Robust Multi-class Graph Transduction* (RMGT) algorithm achieves state-of-the-art performance on a variety of graph transduction tasks. Although RMGT achieves state-of-the-art performance and is parameter-free, this method was specifically designed for using the combinatorial Laplacian within its regularization framework. Unfortunately, the combinatorial Laplacian may not be the most appropriate graph Laplacian for all real applications and recent empirical studies showed that normalized and iterated Laplacians may be better suited than combinatorial Laplacians in general tasks. In this paper, we generalize the RMGT algorithm for any positive semidefinite matrix. Therefore, we provide a novel graph transduction method that can naturally deal with higher order regularization. In order to show the effectiveness of our method, we empirically evaluate it against five state-of-the-art graph-based semi-supervised learning algorithms with respect to graph construction and parameter selection on a number of benchmark data sets. Through a detailed experimental analysis using recently proposed empirical evaluation models, we see that our method achieved competitive performance on most data sets. In addition, our method achieved good stability with respect to the graph's parameter for most data sets and graph construction methods, which is a valuable property for real applications. However, the Laplacian's degree value may have a moderate influence in the performance of our method.**

I. Introduction

Graph-based semi-supervised learning (SSL) algorithms learn from both labeled and unlabeled examples using a weighted graph that encodes similarities between neighbored examples with respect to some distance function. Since these algorithms can use labeled and unlabeled examples together during training stage in order to improve the classifiers' performance, they can be effective in scenarios in which we deal only with a few labeled examples.

Graph-based SSL has gained increased attention in the last few years [1]–[6] because these methods usually perform well when the data lie on a low-dimensional manifold [7], i.e. when the *manifold assumption* holds. Most of these algorithms are based on the optimization of a convex cost function that uses a Laplacian regularizer term as smoothness functional, possibly subject to some fitting and/or normalization constraints. Formally, the Laplacian regularizer term is a smoothness penalty term that tries to reflect the intrinsic geometric structure of the data marginal distribution.

In this paper, we focus on the problem of *graph transduction*. Consequently, we want to generate a classification function from both labeled and unlabeled examples using a weighted graph and (scarce) label information without providing generalization for the entire sample space. In other words, we focus on the SSL algorithms' performance on the set of unlabeled examples. Recently, *de Sousa et al.* [8] provided a comprehensive empirical comparison of graph-based SSL algorithms with respect to graph construction and parameter selection on a number of benchmark data sets. Among their findings, the authors showed that the *Robust Multi-class Graph Transduction* (RMGT) [4] algorithm achieves state-of-the-art performance with respect to graph construction and parameter selection on a variety of these data sets.

RMGT is a *parameter-free* algorithm for graph transduction specifically designed for using the *combinatorial Laplacian* within its regularization framework. Basically, the combinatorial Laplacian is a type of graph Laplacian, which is a matrix operator that, given a nonnegative, symmetric matrix, outputs a *positive semidefinite* (p.s.d.) matrix, i.e., a symmetric matrix with nonnegative eigenvalues. Unfortunately, the combinatorial Laplacian may not be the most appropriate graph Laplacian for all real applications [9]; however, in our experimental analysis we show that the combinatorial Laplacian may be better suited than the normalized Laplacian for text data. In addition, the use of *higher order regularization* can be effective in general SSL tasks [10]. Basically, higher order regularization, in this context, is a type of graph regularization in which the Laplacian's degree value is greater than one. Although the RMGT algorithm achieved state-of-the-art classification performance on a variety of benchmark data sets [8], this method is uncapable to deal with higher order regularization.

In this paper, we provide a novel SSL algorithm, based on the RMGT algorithm, that can naturally deal with higher order regularization. In order to do this, we solve the optimization problem of the RMGT algorithm for any p.s.d. matrix. Consequently, our method can naturally deal with a variety of graph Laplacians, such as normalized and iterated Laplacians, within its regularization framework. Therefore, due to the statistical model in [9] and the theoretical and empirical study in [10], our method can achieve better results than the original RMGT with respect to graph construction in general SSL tasks. Although our method is mathematically simple, we show that it achieves competitive results to state-of-the-art SSL algorithms.

978-1-4799-1959-8/15/$31.00

We call our method *Robust Multi-class Graph Transduction with Higher Order Regularization* (RMGTHOR). By applying a common strategy of constrained optimization to solve the optimization problem of our method, we show that it has a closed-form solution. This closed-form solution can be computed in cubic time in the number of labeled and unlabeled examples, which is the same time complexity of a variety of widely used graph-based SSL algorithms [6], [5], [2]. In practice, our method has a little bit higher running time than the RMGT algorithm. Therefore, we can achieve better classification performance with a little increase in computation time. Moreover, we formally prove that the RMGT algorithm is a special case of our method when we use the combinatorial Laplacian in our method.

Our theoretical findings are supported by a comprehensive empirical comparison between our method and five state-of-the-art SSL algorithms with respect to graph construction and parameter selection on widely used benchmark data sets. Precisely, we evaluate our method against the following SSL algorithms: *Gaussian Fields and Harmonic Functions* (GFHF) [6]; *Local and Global Consistency* (LGC) [5]; *Laplacian Regularized Least Squares* (LapRLS) [2]; *Laplacian Support Vector Machine* (LapSVM) [2]; and RMGT [4]. A brief overview on the regularization framework of these methods will be given in further section.

For a fair empirical comparison, all SSL algorithms were evaluated using a variety of graph construction methods and parameter values. More precisely, the SSL algorithms were evaluated using the same graph construction methods with the same values for the graph's parameter. The values for the regularization parameters of the SSL algorithms were chosen based on the values suggested in the SSL literature [11], [12], [4], [8]. We compare our experimental results with those reported in [1], [8], [10].

In order to provide a comprehensive empirical comparison between our method and state-of-the-art graph-based SSL algorithms with respect to graph construction and parameter selection, we used the empirical evaluation models recently proposed in [8]. Specifically, our experimental analysis is subdivided into the following three parts: (1) *best case analysis*; (2) *evaluation of classifier stability*; and (3) *evaluation of external parameters.*

In the best case analysis, we evaluate the best average error rates of the SSL algorithms obtained over all parameter values. Although this empirical analysis is widely used in the SSL literature [1], it may hide useful information concerning the classifiers' performance with respect to graph construction and parameter selection, being not adequate for comprehensive empirical comparisons [8]. In this analysis, we compare the best performances achieved by our method for each graph construction method with the best performances obtained by the other classifiers over all graph construction methods and parameter values.

In the evaluation of classifier stability, we analyze the performance of the SSL algorithms with respect to graph construction and parameter selection. With this analysis, we want to identify which classifiers provide a good trade-off between performance and stability with respect to the graph's parameter.

In the evaluation of external parameters, we analyze the performance of our method with respect to the graph's parameter and the Laplacian's degree, i.e., we analyze the error surfaces generated by our method with respect to these parameters. With this analysis, we want to verify how stable our method is in this parameter space.

Our results show that our method achieved competitive performances in comparison to the other SSL algorithms in the best case analysis. The results are impressive for the data sets of Gaussian mixtures, in which our method outperformed the competing methods by a large margin. Moreover, our method showed good stability with respect to the graph's parameter on most data sets for many graph construction methods, which is a valuable property for real applications. In many real applications, stable classifiers may be preferable to classifiers that have exceptional performance in only a very narrow range of parameter values [8].

By analyzing the error surfaces generated by our method with respect to the graph's parameter and the Laplacian's degree, we see that our method has good stability with respect to the graph's parameter. However, the Laplacian's degree value may have a moderate influence in the performance of our method in some cases.

The remainder of this paper is organized as follows. Section II provides a background on graph Laplacians, graph-based SSL, and the regularization framework of the RMGT algorithm. Section III formulates our method, providing our main theoretical results. Section IV shows our experimental analysis. Finally, Section V provides our conclusions.

II. Background

Consider a training sample $\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^{n} \subset \mathbb{R}^d$, in which the first l examples are labeled with one of c classes; the remainder $u := n - l$ examples are unlabeled. Let $\mathbb{B} := \{0, 1\}$ and $\mathbb{N}_a := \{i \in \mathbb{N}^* | 1 \leq i \leq a\}$, $\forall a \in \mathbb{N}^*$. Assume that $\boldsymbol{x}_i$, $i \in \mathbb{N}_l$, has label $y_i \in \mathbb{N}_c$. Let $\mathbf{Y}_{\mathcal{L}} \in \mathbb{B}^{l \times c}$ be a label matrix in which $(\mathbf{Y}_{\mathcal{L}})_{ij} = 1$ if and only if $\boldsymbol{x}_i$ has label $y_i = j$. This paper focus on multi-class problems; hence, $\mathbf{Y}_{\mathcal{L}}\mathbf{1}_c = \mathbf{1}_l$ such that $\mathbf{1}_c$ is a c-dimensional 1-entry vector.

All matrices can be subdivided into submatrices of labeled and unlabeled examples. Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ be a weighted matrix generated from the training sample $\mathcal{X}$ and $\mathbf{F} \in \mathbb{R}^{n \times c}$ be the output of a given graph-based SSL algorithm. Assume that $\mathbf{F}$ is subdivided into two submatrices while all other matrices are subdivided into four submatrices. For instance:

$$\mathbf{W} := \begin{bmatrix} \mathbf{W}_{\mathcal{LL}} & \mathbf{W}_{\mathcal{LU}} \\ \mathbf{W}_{\mathcal{UL}} & \mathbf{W}_{\mathcal{UU}} \end{bmatrix} \qquad \mathbf{F} := \begin{bmatrix} \mathbf{F}_{\mathcal{L}} \\ \mathbf{F}_{\mathcal{U}} \end{bmatrix}$$

where $\mathbf{W}_{\mathcal{LL}} \in \mathbb{R}^{l \times l}$ and $\mathbf{F}_{\mathcal{L}} \in \mathbb{R}^{l \times c}$ are the submatrices of $\mathbf{W}$ and $\mathbf{F}$, respectively, on labeled examples, $\mathbf{W}_{\mathcal{UU}} \in \mathbb{R}^{u \times u}$ and $\mathbf{F}_{\mathcal{U}} \in \mathbb{R}^{u \times c}$ are the submatrices of $\mathbf{W}$ and $\mathbf{F}$, respectively, on unlabeled examples, and so on. Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a p.s.d. matrix generated from $\mathbf{W}$; $\mathbf{M}$ is usually some graph Laplacian. In this paper, we consider the problem of how to compute the output matrix $\mathbf{F}_{\mathcal{U}} \in \mathbb{R}^{u \times c}$ given $\mathbf{M}$ and the label matrix $\mathbf{Y}_{\mathcal{L}}$ in a parameter-free way by solving the optimization problem of the RMGT algorithm using $\mathbf{M}$ in the Laplacian regularizer term.

A. Graph Laplacians

The graph Laplacians are important tools for many fields of machine learning, such as *spectral clustering* and *dimensionality reduction* [13]. One of the motivations for using a graph Laplacian in machine learning (and specially in SSL) is that it induces an adaptive regularization functional that may adapt to the data's density and geometric structure [14].

Given a (sparse) weighted matrix $\mathbf{W}$, we can compute a graph Laplacian in order to start the diffusion process. The *combinatorial* Laplacian is defined by $\mathbf{\Delta} := \mathbf{D} - \mathbf{W}$ where $\mathbf{D} := \operatorname{diag}(\mathbf{W}\mathbf{1}_n)$. The *normalized* Laplacian is defined by $\mathbf{L} := \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ where $\mathbf{I}_n$ is the n-by-n identity matrix. We see that if $\mathbf{W} \geq 0$ and $\mathbf{W} = \mathbf{W}^\top$, $\mathbf{\Delta}$ and $\mathbf{L}$ are p.s.d. matrices ($\mathbf{\Delta}, \mathbf{L} \succcurlyeq 0$); hence, the objective functions of most graph-based SSL algorithms become convex. The superscript $^\top$ represents transposition of matrices or vectors.

Although the combinatorial and normalized Laplacians are the most commonly used graph Laplacians in the SSL literature [15], [1], there are other graph Laplacians that can also be effectively applied in graph-based SSL as well [14], [13]. For instance, the *iterated* Laplacian is defined by $\mathbf{L}_i := \mathbf{L}_b^p$ in which $\mathbf{L}_b \in \mathbb{R}^{n\times n}$ is the "basis" Laplacian and $p \in \mathbb{N}^*$ is the Laplacian's degree. In practice, $\mathbf{L}_b$ can be any graph Laplacian. Therefore, we can easily apply iterated Laplacians in the regularization framework of graph-based SSL algorithms, achieving the same closed-form solutions.

B. Graph-based SSL

Given a graph Laplacian constructed from the weighted matrix $\mathbf{W}$, most graph-based SSL algorithms generates the output matrix $\mathbf{F}$ by minimizing a (usually convex) cost function using a Laplacian regularizer term. For instance, consider the following optimization problem:

$$\min_{\mathbf{F}\in\mathbb{R}^{n\times c}} \operatorname{tr}\left(\mathbf{F}^\top\mathbf{M}\mathbf{F} + (\mathbf{F}-\mathbf{Y})^\top\mathbf{\Sigma}(\mathbf{F}-\mathbf{Y})\right) \tag{1}$$

where $\mathbf{Y} := \begin{bmatrix} \mathbf{Y}_\mathcal{L} \\ \mathbf{O}_{u\times c} \end{bmatrix} \in \mathbb{B}^{n\times c}$ is the "extended" label matrix, $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with nonnegative values, and $\mathbf{O}_{u\times c}$ is the $u \times c$ null matrix. The symbol $\operatorname{tr}(\cdot)$ stands for the trace matrix operator. If $\mathbf{M} = \mathbf{L}$ and $\mathbf{\Sigma}_{ii} = \mu > 0$, $\forall i \in \mathbb{N}_n$, we have the LGC algorithm [5]; if $\mathbf{M} = \mathbf{\Delta}$, $\mathbf{\Sigma}_{ii} \to +\infty$, $\forall i \in \mathbb{N}_l$, and $\mathbf{\Sigma}_{ii} = 0$, $\forall i \in \mathbb{N}_n \backslash \mathbb{N}_l$, we have the GFHF algorithm [6] (see [16]).

The *manifold regularization* [2] framework can be viewed as the following optimization problem:

$$\min_{f\in\mathcal{H}_\mathcal{K}} \frac{1}{l}\sum_{i=1}^{l} \mathcal{V}(\boldsymbol{x}_i, y_i, f) + \gamma_A\|f\|_{\mathcal{H}_\mathcal{K}} + \gamma_I\mathbf{f}^\top\mathbf{M}\mathbf{f} \tag{2}$$

where $\mathcal{V}(\boldsymbol{x}_i, y_i, f)$ is a cost function, $\mathcal{H}_\mathcal{K}$ is the *Reproducing Kernel Hilbert Space* (*RKHS*) for the kernel $\mathcal{K}$, $\|\cdot\|_{\mathcal{H}_\mathcal{K}}$ is the norm in $\mathcal{H}_\mathcal{K}$, $\mathbf{f} := [f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)]^\top \in \mathbb{R}^n$, γ_A and γ_I are the regularization parameters, and $y_i \in \{-1, +1\}$, $\forall i \in \mathbb{N}_l$. If $\mathcal{V}(\boldsymbol{x}_i, y_i, f) = (y_i - f(\boldsymbol{x}_i))^2$, we have the LapRLS algorithm [2]; if $\mathcal{V}(\boldsymbol{x}_i, y_i, f) = \max(0, 1 - y_i f(\boldsymbol{x}_i))$, we have the LapSVM algorithm [2].

C. The RMGT algorithm

The RMGT algorithm [4] is based on the GFHF algorithm, incorporating two normalization constraints that encode output normalization and class prior knowledge. The multi-class version of the GFHF algorithm can be viewed as the following optimization problem:

$$\min_{\mathbf{F}\in\mathbb{R}^{n\times c}} \operatorname{tr}\left(\mathbf{F}^\top\mathbf{\Delta}\mathbf{F}\right) \quad \text{s.t.} \quad \mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L} \tag{3}$$

Considering $\mathbf{\Delta} \succcurlyeq 0$, the optimization problem in (3) is convex in $\mathbf{F}$. As shown in [6], the closed-form solution of (3) is given by $\mathbf{F}_\mathcal{U} = -\mathbf{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{\Delta}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_\mathcal{L}$. In the RMGT framework, we optimize the objective function in (3) s.t. $\mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L}$ and the following normalization constraints: (1) $\mathbf{F}\mathbf{1}_c = \mathbf{1}_n$; and (2) $\mathbf{F}^\top\mathbf{1}_n = n\boldsymbol{\omega}$, in which $\boldsymbol{\omega} \in \mathbb{R}^c$ is considered a model's parameter[1]. Mathematically, the optimization framework of the RMGT algorithm is given by:

$$\begin{aligned} &\min_{\mathbf{F}\in\mathbb{R}^{n\times c}} \operatorname{tr}\left(\mathbf{F}^\top\mathbf{\Delta}\mathbf{F}\right) \\ &\text{s.t.} \quad \mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L}, \;\; \mathbf{F}\mathbf{1}_c = \mathbf{1}_n, \;\; \mathbf{F}^\top\mathbf{1}_n = n\boldsymbol{\omega} \end{aligned} \tag{4}$$

Since we do not impose that $\mathbf{F} \geq 0$, the constraint $\mathbf{F}\mathbf{1}_c = \mathbf{1}_n$ enforces the soft labels in $\mathbf{F}$ to act like *pseudo-probabilities*. The constraint $\mathbf{F}^\top\mathbf{1}_n = n\boldsymbol{\omega}$ enforces output normalization using class prior knowledge. *Liu & Chang* [4] showed that (4) has the following closed-form solution:

$$\mathbf{F}_\mathcal{U} = \mathbf{F}_\mathcal{U}^{(0)} + \frac{\mathbf{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}{\mathbf{1}_u^\top\mathbf{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}\left(n\boldsymbol{\omega}^\top - \mathbf{1}_l^\top\mathbf{Y}_\mathcal{L} - \mathbf{1}_u^\top\mathbf{F}_\mathcal{U}^{(0)}\right) \tag{5}$$

where $\mathbf{F}_\mathcal{U}^{(0)} = -\mathbf{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{\Delta}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_\mathcal{L}$ is the closed-form solution of the GFHF algorithm.

III. Proposed Method

In order to formulate our method, we consider the same optimization problem in (4) changing the combinatorial Laplacian $\mathbf{\Delta}$ in the objective function for any p.s.d. matrix $\mathbf{M}$. We provide a mathematical formulation of our method without assuming any other special property of $\mathbf{M}$. Given a p.s.d. matrix $\mathbf{M}$ and a label matrix $\mathbf{Y}_\mathcal{L}$ such that $\mathbf{Y}_\mathcal{L}\mathbf{1}_c = \mathbf{1}_l$, our method computes the output matrix $\mathbf{F}_\mathcal{U}$ via convex regularization in a parameter-free way. In practice, our method generalizes the RMGT algorithm for any p.s.d. matrix and we can compute its closed-form solution in $\mathcal{O}\left(n^3\right)$ time with $\mathcal{O}\left(n^2\right)$ space complexities, which are the same time and space complexities of a variety of widely used graph-based SSL algorithms [2], [4]–[6].

The main theoretical result in this paper is given in Theorem 1, in which we provide the closed-form solution of our method by applying a standard strategy of constrained optimization. In Corollary 1, we show that our method is reduced to the RMGT algorithm when $\mathbf{M} = \mathbf{\Delta}$.

[1] Empirically speaking, $\boldsymbol{\omega}$ can be the class prior probabilities or the uniform class distribution ($\boldsymbol{\omega} = \mathbf{1}_c/c$), as suggested in [4]. $\boldsymbol{\omega}$ is *not* a parameter for tunning.

Mathematically, our method solves the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{F}\in\mathbb{R}^{n\times c}} \; & \operatorname{tr}\left(\mathbf{F}^\top\mathbf{M}\mathbf{F}\right) \\ \text{s.t.} \quad & \mathbf{F}_{\mathcal{L}} = \mathbf{Y}_{\mathcal{L}}, \;\; \mathbf{F}\mathbf{1}_c = \mathbf{1}_n, \;\; \mathbf{F}^\top\mathbf{1}_n = n\boldsymbol{\omega} \end{aligned} \tag{6}$$

If $\mathbf{M} \succcurlyeq 0$, then $\mathbf{M} = \mathbf{M}^\top$; hence, $\mathbf{M}_{\mathcal{U}\mathcal{L}} = \mathbf{M}_{\mathcal{L}\mathcal{U}}^\top$. Therefore, we have:

$$\operatorname{tr}\left(\mathbf{Y}_{\mathcal{L}}^\top\mathbf{M}_{\mathcal{L}\mathcal{U}}\mathbf{F}_{\mathcal{U}}\right) = \operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}}\right) \tag{7}$$

Considering the hard constraint $\mathbf{F}_{\mathcal{L}} = \mathbf{Y}_{\mathcal{L}}$ and Eq. (7), we obtain:

$$\begin{aligned} \operatorname{tr}\left(\mathbf{F}^\top\mathbf{M}\mathbf{F}\right) = & \operatorname{tr}\left(\mathbf{Y}_{\mathcal{L}}^\top\mathbf{M}_{\mathcal{L}\mathcal{L}}\mathbf{Y}_{\mathcal{L}}\right) + \operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}\mathbf{F}_{\mathcal{U}}\right) \\ & + 2\operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}}\right) \end{aligned}$$

Then, the optimization problem in (6) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{F}_{\mathcal{U}}\in\mathbb{R}^{u\times c}} \; & \operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}\mathbf{F}_{\mathcal{U}}\right) + 2\operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}}\right) \\ \text{s.t.} \quad & \mathbf{F}_{\mathcal{U}}\mathbf{1}_c = \mathbf{1}_u, \;\; \mathbf{F}_{\mathcal{U}}^\top\mathbf{1}_u = n\boldsymbol{\omega} - \mathbf{Y}_{\mathcal{L}}^\top\mathbf{1}_l \end{aligned} \tag{8}$$

If $\mathbf{M} \succcurlyeq 0$, then $\mathbf{M}_{\mathcal{U}\mathcal{U}} \succcurlyeq 0$. Therefore, the objective function of the optimization problem in (8) is convex in $\mathbf{F}_{\mathcal{U}}$. We provide the closed-form solution of (8) in the following theorem.

Theorem 1: The closed-form solution of the optimization problem in (8) *is given by*

$$\begin{aligned} \mathbf{F}_{\mathcal{U}} = & -\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}} + \frac{\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}\tau \\ & + \frac{1}{c}\left(\mathbf{1}_u + \mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l - \nu\right)\mathbf{1}_c^\top \end{aligned} \tag{9}$$

in which

$$\begin{aligned} \tau &= n\boldsymbol{\omega}^\top - \mathbf{1}_l^\top\mathbf{Y}_{\mathcal{L}} + \mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}} \\ \nu &= \frac{\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}\left(u + \mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l\right) \end{aligned}$$

Proof: The Lagrangian corresponding to the optimization problem in (8) is given by

$$\begin{aligned} \mathcal{L}(\mathbf{F}_{\mathcal{U}}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = & \operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}\mathbf{F}_{\mathcal{U}}\right) + 2\operatorname{tr}\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}}\right) \\ & - \boldsymbol{\xi}^\top\left(\mathbf{F}_{\mathcal{U}}\mathbf{1}_c - \mathbf{1}_u\right) \\ & - \boldsymbol{\lambda}^\top\left(\mathbf{F}_{\mathcal{U}}^\top\mathbf{1}_u - n\boldsymbol{\omega} + \mathbf{Y}_{\mathcal{L}}^\top\mathbf{1}_l\right) \end{aligned}$$

where $\boldsymbol{\xi} \in \mathbb{R}^u$ and $\boldsymbol{\lambda} \in \mathbb{R}^c$ are the Lagrange multipliers. Zeroing $\partial\mathcal{L}/\partial\mathbf{F}_{\mathcal{U}}$, we obtain:

$$\mathbf{F}_{\mathcal{U}} = -\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}} + \frac{1}{2}\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\boldsymbol{\xi}\mathbf{1}_c^\top + \frac{1}{2}\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u\boldsymbol{\lambda}^\top \tag{10}$$

Substituting (10) in the constraint $\mathbf{F}_{\mathcal{U}}\mathbf{1}_c = \mathbf{1}_u$, we obtain:

$$c\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\boldsymbol{\xi} + \mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u\boldsymbol{\lambda}^\top\mathbf{1}_c = 2\mathbf{1}_u + 2\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l \tag{11}$$

Substituting (10) in the constraint $\mathbf{F}_{\mathcal{U}}^\top\mathbf{1}_u = n\boldsymbol{\omega} - \mathbf{Y}_{\mathcal{L}}^\top\mathbf{1}_l$, we obtain:

$$\boldsymbol{\lambda}^\top = \frac{2}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}\left(n\boldsymbol{\omega}^\top - \mathbf{1}_l^\top\mathbf{Y}_{\mathcal{L}} + \boldsymbol{\lambda}^{(0)}\right) \tag{12}$$

in which

$$\boldsymbol{\lambda}^{(0)} = \mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_{\mathcal{L}} - \frac{1}{2}\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\boldsymbol{\xi}\mathbf{1}_c^\top$$

Substituting (12) in (11), we obtain:

$$\boldsymbol{\xi} = \frac{2}{c}\left(\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1} - \frac{\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}\right)^{-1}\boldsymbol{\xi}^{(0)} \tag{13}$$

where

$$\begin{aligned} \boldsymbol{\xi}^{(0)} = & \mathbf{1}_u + \mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l - \frac{u\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u} \\ & - \frac{\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{M}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l}{\mathbf{1}_u^\top\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u} \end{aligned}$$

Substituting (12) and (13) in (10), we get (9). This finishes the proof. ■

Corollary 1: If $\mathbf{M} = \boldsymbol{\Delta}$, (9) *is reduced to* (5).

Proof: Let $\mathbf{v} \in \mathbb{R}^u$ such that

$$\begin{aligned} \mathbf{v} = & \mathbf{1}_u + \boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l - \frac{u\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u}{\mathbf{1}_u^\top\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u} \\ & - \frac{\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u\mathbf{1}_u^\top\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l}{\mathbf{1}_u^\top\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}^{-1}\mathbf{1}_u} \end{aligned}$$

We know that $\boldsymbol{\Delta}\mathbf{1}_n = \mathbf{0}_n$; hence, $\boldsymbol{\Delta}_{\mathcal{U}\mathcal{L}}\mathbf{1}_l = -\boldsymbol{\Delta}_{\mathcal{U}\mathcal{U}}\mathbf{1}_u$. Then, $\mathbf{v} = \mathbf{0}_u$. Substituting $\mathbf{v}$ in (9), we get (5). This finishes the proof. ■

In Theorem 1 and Corollary 1, we showed the main theoretical results in this paper. Although our method can be considered mathematically simple because we used standard techniques of constrained optimization in order to solve (8), we show through a comprehensive experimental analysis on benchmark data sets that our method achieves competitive (or even better) performance to (than) state-of-the-art methods for graph-based SSL with respect to graph construction and parameter selection.

TABLE I. AVERAGE ERROR RATES (%) AND STANDARD DEVIATIONS (%) OF THE SSL ALGORITHMS AND THEIR AVERAGE RANKINGS IN THE PARTITIONS OF 10 LABELED EXAMPLES.

	USPS	COIL$_2$	DIGIT-1	G-241N	G-241C	TEXT	ranking
GFHF (best)	9.75(4.50)	35.07 (3.82)	9.35 (4.51)	46.12 (7.61)	46.19 (7.25)	37.51 (6.85)	11.75
LGC (best)	9.93 (4.34)	34.81 (6.22)	10.47 (4.66)	37.95 (6.66)	40.10 (5.46)	34.78 (6.55)	10.167
LapRLS (best)	9.75(4.53)	31.78 (7.81)	9.33 (4.48)	38.06 (6.52)	40.11 (6.06)	34.58 (6.14)	9.417
LapSVM (best)	9.91 (2.51)	31.54 (6.24)	8.67 (3.89)	38.90 (6.50)	40.82 (6.66)	37.49 (7.07)	10
RMGT (best)	13.08 (3.41)	28.00 (4.67)	7.50 (2.43)	42.75 (7.33)	38.31 (6.02)	27.77 (5.95)	7.667
Chap	13.61	-	5.44	18.64	22.76	27.15	-
IterLap	13.10	-	6.54	20.99	18.01	38.84	-
RMGTHOR-symKNN-RBF	13.32 (3.64)	30.98 (3.52)	7.42 (3.37)	36.11 (16.00)	26.94 (4.95)	30.22 (2.47)	7.167
RMGTHOR-mutKNN-RBF	10.88 (4.12)	27.80 (5.10)	6.38 (2.37)	36.83 (12.13)	33.26 (4.41)	28.74 (4.02)	5.167
RMGTHOR-symFKNN-RBF	12.54 (3.32)	30.89 (3.53)	7.00 (2.74)	36.86 (15.34)	26.79 (4.94)	29.42 (2.31)	6.667
RMGTHOR-symKNN-HM	14.51 (4.87)	28.74 (6.52)	6.03 (1.96)	35.95 (15.70)	**26.59 (4.53)**	30.31 (3.23)	5.667
RMGTHOR-mutKNN-HM	**10.60 (4.18)**	27.69 (5.58)	**5.22 (1.77)**	**35.20 (12.80)**	34.12 (4.51)	29.00 (3.57)	**3.333**
RMGTHOR-symFKNN-HM	13.78 (4.69)	28.34 (7.88)	5.89 (2.00)	36.62 (15.29)	26.76 (4.96)	28.65 (2.68)	4.5
RMGTHOR-symKNN-LLE	14.77 (4.11)	29.79 (5.70)	6.30 (2.02)	41.54 (9.67)	36.14 (3.92)	32.13 (3.56)	9.333
RMGTHOR-mutKNN-LLE	10.69 (3.36)	**27.30 (5.56)**	5.46 (1.84)	41.38 (6.59)	39.84 (5.05)	**28.71 (3.13)**	5.333
RMGTHOR-symFKNN-LLE	14.22 (4.12)	28.93 (4.89)	6.15 (2.07)	41.82 (10.06)	36.37 (5.17)	30.60 (3.62)	8.833

IV. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate our method against the GFHF, LGC, LapRLS, LapSVM, and RMGT algorithms in transductive learning tasks with respect to graph construction and parameter selection on six benchmark data sets, available in [1], using the preprocessing step described in [8][2]. In order to compare our results with those reported in [1], [8], [10], [17], we used the same graph construction methods and parameter setting chosen in [8] and the data splits of 10 and 100 labeled examples suggested in [1]. Due to reasons concerning reproducibility, the source code used in our experiments as well as our experimental results are freely available[3].

A. Experimental setup

Let $\Psi : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ be a distance function. In order to compute a distance matrix $\mathbf{\Psi} \in \mathbb{R}^{n \times n}$ from the training sample $\mathcal{X}$ such that $\mathbf{\Psi}_{ij} = \Psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we chose the cosine distance for the TEXT data set and the L_2-norm for the other data sets. From $\mathbf{\Psi}$, we generated three adjacency graphs based on the k-nearest neighbors (kNN) graph: *symmetric* kNN (symKNN); *mutual* kNN (mutKNN); and *symmetry favored* kNN (symFKNN) [4]. Let $\mathbf{A}_k \in \mathbb{B}^{n \times n}$ be the adjacency matrix of the kNN graph. The symKNN graph has adjacency matrix $\mathbf{A}_k^{(sym)} = \max\left(\mathbf{A}_k, \mathbf{A}_k^\top\right)$; the mutKNN graph has adjacency matrix $\mathbf{A}_k^{(mut)} = \min\left(\mathbf{A}_k, \mathbf{A}_k^\top\right)$; the symFKNN graph has adjacency matrix $\mathbf{A}_k^{(symF)} = \mathbf{A}_k + \mathbf{A}_k^\top$. The parameter k was chosen in the set $\{4, 6, 8, \cdots, 40\}$.

In order to generate a sparse, symmetric, weighted matrix $\mathbf{W}$ from the adjacency matrix[4] $\mathbf{A} \in \{0, 1, 2\}^{n \times n}$ and the distance matrix $\mathbf{\Psi}$, we used the following three weighted matrix generation methods: RBF kernel; similarity function of *Hein & Maier* (HM) [18]; and *Locally Linear Embedding* (LLE) [19]. The HM similarity function can be viewed as an RBF kernel with adaptive kernel sizes; the LLE method generates $\mathbf{W}$ via local reconstruction minimization on each example $\boldsymbol{x}_i$.

[2]http://sites.labic.icmc.usp.br/sousa/experiments_graph_SSL/.

[3]http://sites.labic.icmc.usp.br/sousa/constrained_graph_SSL/.

[4]Note that the symFKNN graph has a non-binary adjacency matrix.

Since it may not be straightforward to choose an adequate value for the RBF kernel's parameter σ when we have only a few labeled examples, we estimated its value as $\sigma = \frac{1}{3n}\sum_{i=1}^{n} \Psi\left(\boldsymbol{x}_i, \boldsymbol{x}_{i_k}\right)$ where $\boldsymbol{x}_{i_k}$ is the k-th nearest neighbor of $\boldsymbol{x}_i$, as suggested in [11]. We used in our experimental analysis the following weighted graphs: symKNN-RBF (G1); mutKNN-RBF (G2); symFKNN-RBF (G3); symKNN-HM (G4); mutKNN-HM (G5); symFKNN-HM (G6); symKNN-LLE (G7); mutKNN-LLE (G8); and symFKNN-LLE (G9).

From the weighted matrix $\mathbf{W}$, we have to generate a graph Laplacian in order to start the diffusion process. In an attempt to avoid numerical issues while running our method (specifically when computing $\mathbf{M}_{\mathcal{U}\mathcal{U}}^{-1}$), we generated the combinatorial Laplacian as $\mathbf{\Delta} = \eta\mathbf{D} - \mathbf{W}$ and the normalized Laplacian as $\mathbf{L} = \eta\mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ with $\eta > 1$, as suggested in [8]. In our experiments, we set $\eta = 1.01$.

Since higher order regularization may be effective on SSL [10], we used the iterated Laplacian in our method. Since the normalized Laplacian $\mathbf{L}$ performs better than the combinatorial Laplacian $\mathbf{\Delta}$ in general SSL tasks [9], we chose the normalized Laplacian as the "basis" Laplacian. The Laplacian's degree value was chosen in the set $\{1, 2, 3, 4, 5\}$.

B. Best case analysis

Tables I and II show the best performances (average error rates) of our method for each graph construction method as well as the best performances for the other classifiers over all graph construction methods in the partitions of 10 and 100 labeled examples, respectively. All the competing SSL algorithms were evaluated in the same nine (G1-G9) weighted graphs with the same values for the parameter k (extensive results for these classifiers can be found in [8], [17]). However, due to lack of space, we only report in this paper the best result over all graph construction methods and parameter values in each data set for the GFHF, LGC, LapRLS, LapSVM, and RMGT algorithms. The best results in [1] are reported as "Chap" while the best results in [10] are reported as "IterLap". Since the authors in [1], [10] only reported the error rates of the methods they were analyzing, we do not report the standard deviations for Chap and IterLap.

TABLE II. AVERAGE ERROR RATES (%) AND STANDARD DEVIATIONS (%) FOR THE SSL ALGORITHMS AND THEIR AVERAGE RANKINGS IN THE PARTITIONS OF 100 LABELED EXAMPLES.

	USPS	$COIL_2$	DIGIT-1	G-241N	G-241C	TEXT	ranking
GFHF (best)	3.26 (1.00)	1.09 (1.59)	2.01 (0.58)	31.71 (6.08)	37.59 (5.75)	23.69 (2.48)	9.917
LGC (best)	3.54 (0.96)	1.09 (1.59)	2.13 (0.72)	22.04 (1.75)	29.81 (2.57)	23.35 (2.26)	9.75
LapRLS (best)	3.17(0.93)	0.35(0.75)	1.87(0.51)	21.89 (1.73)	28.73 (3.43)	23.48 (1.64)	6.167
LapSVM (best)	3.71 (1.44)	0.36 (0.74)	1.92 (0.48)	21.64 (3.90)	25.51 (5.58)	22.43 (1.11)	**5.667**
RMGT (best)	4.71 (2.25)	0.85 (0.80)	1.95 (0.40)	26.01 (8.82)	25.11 (2.12)	20.76(1.18)	7.25
Chap	4.68	-	2.44	4.95	13.49	23.09	-
IterLap	3.96	-	2.22	10.55	14.82	25.77	-
RMGTHOR-symKNN-RBF	3.80 (1.05)	3.10 (1.79)	2.14 (0.40)	9.95 (0.88)	18.56 (0.93)	21.17 (0.92)	7.333
RMGTHOR-mutKNN-RBF	3.31 (0.85)	2.58 (1.43)	2.17 (0.31)	15.82 (1.17)	26.48 (1.99)	21.10 (0.90)	7.667
RMGTHOR-symFKNN-RBF	3.67 (1.09)	2.93 (1.90)	2.15 (0.37)	10.56 (0.87)	19.40 (1.09)	**21.09 (0.97)**	7.0
RMGTHOR-symKNN-HM	4.38 (0.95)	2.91 (1.17)	**1.93 (0.41)**	**9.63 (0.94)**	**18.23 (0.83)**	22.54 (1.22)	6.667
RMGTHOR-mutKNN-HM	**3.18 (0.66)**	2.46 (1.46)	2.04 (0.34)	15.80 (1.48)	27.29 (1.98)	22.19 (0.83)	7.0
RMGTHOR-symFKNN-HM	4.04 (0.88)	2.26 (1.03)	1.97 (0.53)	9.64 (0.79)	18.79 (0.93)	22.05 (0.98)	6.167
RMGTHOR-symKNN-LLE	4.21 (0.65)	0.80 (0.69)	2.08 (0.41)	20.91 (1.59)	26.45 (1.28)	22.87 (1.17)	8.667
RMGTHOR-mutKNN-LLE	3.86 (0.87)	**0.38 (0.49)**	2.06 (0.30)	23.67 (1.30)	32.41 (2.18)	21.94 (0.78)	8.667
RMGTHOR-symFKNN-LLE	3.75 (0.44)	0.61 (0.60)	1.95 (0.41)	21.77 (1.19)	28.39 (1.98)	22.11 (0.86)	7.083

Note that we report in Tables I and II the best performances of our method for *each* graph construction method on each data set; for the other SSL algorithms, we only report their best performance over *all* graph construction methods, due to lack of space. Therefore, the results in these tables may appear slightly pessimistic to our method. The best result of our method in each data set over all graph construction methods is marked in bold face while the best overall result in each data set is boxed. The four worst results in each data set have a grey background. We also report the average rankings of the algorithms over all data sets. Since the results for the $COIL_2$ data set do not appear in [1], [10], Chap and IterLap were not considered during ranking computation.

In Table I, we see that our method achieved the best overall results in 2 out of 6 data sets as well as competitive results to the other SSL algorithms in the G-241N, G-241C, and TEXT data sets for all graph construction methods. The results are impressive in the G-241C data set, in which our method outperformed the competing methods by a large margin when applying the graphs generated by the HM method and the RBF kernel. However, our method did not outperform the best results reported in [1], [10]. We also see that our method showed competitive results in the USPS data set only when we apply the graphs generated by mutKNN. By analyzing the average rankings, we see that our method outperformed the best overall average rankings for the other SSL algorithms for most graph construction methods.

In Table II, we see that our method achieved competitive results to the other SSL algorithms in the USPS, $COIL_2$, DIGIT-1, and TEXT data sets for all graph construction methods as well as good to exceptional results in the G-241N and G-241C data sets in many settings. Moreover, our method outperformed the results of Chap and IterLap in the DIGIT-1 and TEXT data sets. Since the RMGT algorithm achieved better results than our method in the TEXT data set, we suppose that the combinatorial Laplacian may be better suited than the normalized Laplacian for text data. Using our method on the G-241C and G-241N data sets, the graphs generated by mutKNN achieved worse performance than those generated by symKNN and symFKNN. In addition, the graphs generated by the HM method and the RBF kernel achieved better results than those generated by the LLE method.

TABLE III. AVERAGE RANKINGS OF THE SSL ALGORITHMS WITH RESPECT TO GRAPH CONSTRUCTION ON THE PARTITIONS OF 10 LABELED EXAMPLES.

	G1	G2	G3	G4	G5	G6
GFHF	4.833	5.083	4.667	5.333	5.5	5.333
LGC	3.833	4.583	3.667	3.667	4.667	4
LapRLS	3.167	2.5	3.667	3.5	2.833	3.5
LapSVM	4.333	3.5	4.167	4.167	3.5	4
RMGT	3.167	3.5	3.167	3.167	3.167	3.167
RMGTHOR	**1.667**	**1.833**	**1.667**	**1.167**	**1.333**	**1**

	G7	G8	G9	mean
GFHF	5.5	5.667	5.417	5.259
LGC	4.333	4.5	4.167	4.157
LapRLS	2.833	2.833	3.417	3.139
LapSVM	3.167	3	3	3.648
RMGT	3.5	3.5	3.5	3.315
RMGTHOR	**1.667**	**1.5**	**1.5**	**1.482**

TABLE IV. AVERAGE RANKINGS OF THE SSL ALGORITHMS WITH RESPECT TO GRAPH CONSTRUCTION ON THE PARTITIONS OF 100 LABELED EXAMPLES.

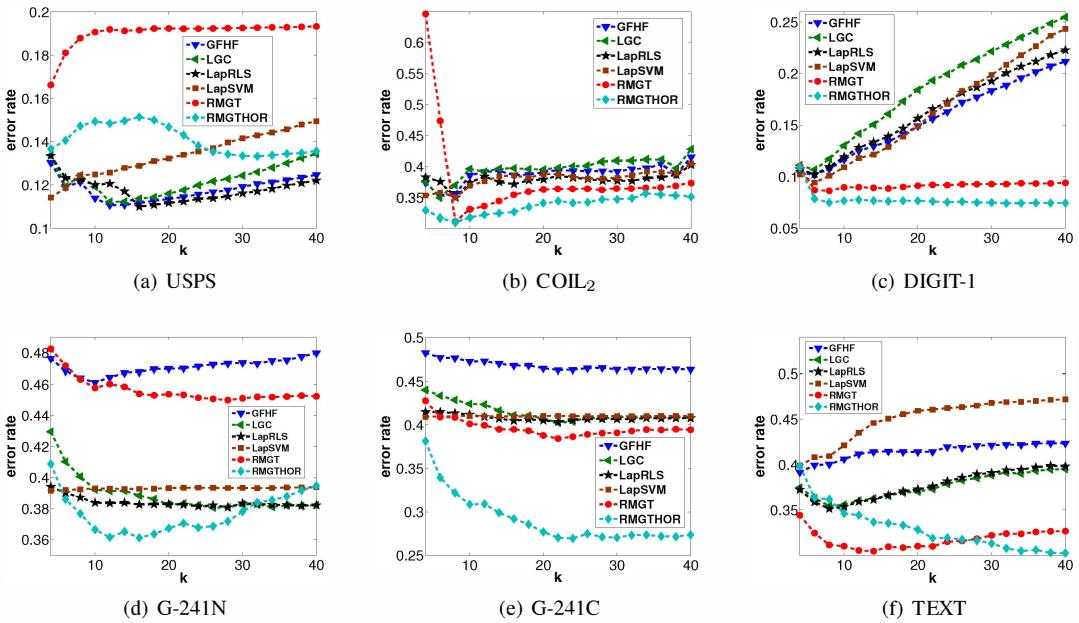
	G1	G2	G3	G4	G5	G6
GFHF	4.583	4.75	4.583	5.5	5.167	5.5
LGC	4.25	4.417	4.25	4.833	4.667	4.833
LapRLS	**2.583**	**2.333**	**2.583**	2.667	2.667	3.167
LapSVM	**2.583**	2.5	**2.583**	3.167	**1.833**	2.667
RMGT	4.167	4.333	4.167	3.5	4.333	3.25
RMGTHOR	2.833	2.667	2.833	**1.333**	2.333	**1.583**

	G7	G8	G9	mean
GFHF	5.5	5.5	5.5	5.176
LGC	5.0	5.0	5.0	4.694
LapRLS	3.167	2.5	2.833	2.370
LapSVM	2.833	2.333	3.0	2.611
RMGT	3.167	3.833	3.333	3.787
RMGTHOR	**1.333**	**1.833**	**1.333**	**2.009**

Tables III and IV show the average rankings of the SSL algorithms with respect to graph construction. The best average ranking for each graph construction method is marked in bold face. In order to statistically evaluate our results, we ran the Friedman test[5] with the Bonferroni correction using a significance level of 0.05. We applied the software Orange[6] to run this statistical test using the average rankings shown in Tables III and IV. The rankings for the algorithms that were statistically outperformed by our method are marked with a grey background.

[5] See [20] for a review on statistical tests for machine learning.

[6] http://orange.biolab.si/



(a) USPS (b) COIL$_2$ (c) DIGIT-1

(d) G-241N (e) G-241C (f) TEXT

Fig. 1. Average error rates of the SSL algorithms with respect to k using the symKNN-RBF graph in the partitions of 10 labeled examples.

In Table III, we see that our method achieved the best average ranking for *all* graph construction methods as well as the best overall average ranking. The results are impressive for the symFKNN-HM (G6) graph, in which our method achieved the best performance in *all* data sets.

In Table IV, we see that our method achieved the best average ranking for the graphs generated by the LLE method and for the symKNN-HM (G4) and symFKNN-HM (G6) graphs. Moreover, our method achieved the best overall average ranking, being the most appropriate SSL algorithm in terms of average ranking given a weighted graph. However, for the graphs generated by the RBF kernel, our method was outperformed by the LapRLS and LapSVM algorithms. Although our method frequently outperformed the competing methods with respect to graph construction in terms of average ranking (as shown in Tables III and IV), the statistical test found significant differences only in a few cases. This is likely to be due to the use of a non-parametric statistical test which requires additional evidence (results in more data sets) in order to detect significant differences.

C. Evaluation of classifier stability

Fig. 1 shows the performances of the SSL algorithms in the partitions of 10 labeled examples with respect to k using the symKNN-RBF graph, which is widely used in the SSL literature [1]. In Fig. 1, we see that our method achieved an outstanding performance with respect to k on most data sets. In addition, our method achieved the best performance for a significant range of k values for all data sets, excluding USPS and TEXT. In the COIL$_2$, DIGIT-1, and G-241C data sets, our method achieved the lowest average error rates for virtually the entire range of k values. We note that no other single algorithm achieved similar performance. On these same three data sets, the runner-up method was the RMGT algorithm. However, the RMGT algorithm showed high instability on the COIL$_2$ data set when $k \in \{4, 6\}$.

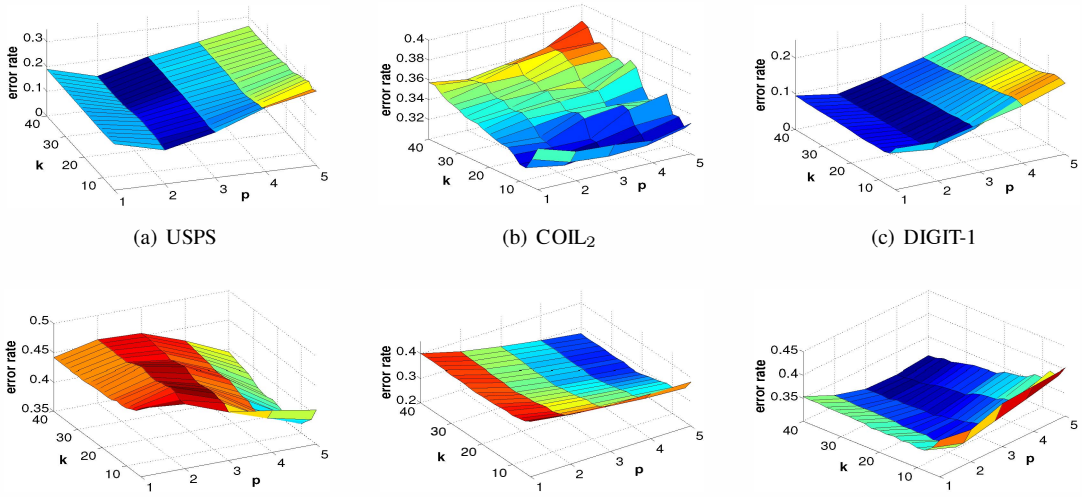
In the G-241C data set, the competing methods achieved a slightly better stability than our method. However, our method outperformed the other SSL algorithms by a large margin in this data set for most values of k. In the DIGIT-1 data set, our method and the RMGT algorithm achieved exceptional performance and stability while the other SSL algorithms showed worse performance and frequently higher instability.

In the G-241N data set, our method significantly improved the RMGT's performance, achieving competitive results to the other classifiers. Unfortunately, our method achieved worse performance than the RMGT algorithm in the TEXT data set for sparse graphs. In the USPS data set, our method also significantly improved the RMGT's performance. However, this improvement was not enough to provide competitive performance to the other classifiers.

D. Evaluation of external parameters

Fig. 2 shows the error surfaces for our method using the symKNN-RBF graph with respect to k and p in the partitions of 10 labeled examples. We see that our method achieved good stability with respect to k in the USPS, DIGIT-1, G-241C, and TEXT data sets, which is a valuable property for real applications. In contrast, the Laplacian's degree value showed a moderate influence in the performance of our method.

The error surfaces for the USPS and DIGIT-1 data sets showed similar patterns, with the optimal results occurring when $p = 2$. For the COIL$_2$ data set, the value of p showed small influence in the performance of our method while the value of k showed a moderate influence in its performance when $p = 5$. For the TEXT data set, the best results were achieved when we applied our method on dense graphs (specifically $k \geq 20$) with $p \geq 3$. We note that our method achieved good to exceptional performances when $p = 2$ using sparse graphs in all data sets, excluding the *artificial* data sets G-241C and G-241N.



(a) USPS (b) COIL_2 (c) DIGIT-1

(d) G-241N (e) G-241C (f) TEXT

Fig. 2. Error surfaces of our method with respect to k and p using the symKNN-RBF graph in the partitions of 10 labeled examples.

By analyzing all the other results (not shown here due to lack of space), we see that our method achieved its best performance in the USPS, DIGIT-1, COIL_2, and TEXT data sets when $p = 2$ for most graph construction methods when we use $k \simeq 10$. However, in the *artificial* data sets G-241C and G-241N, $p = 5$ is the most appropriate value for the Laplacian's degree for most graph construction methods while $k \simeq 40$ is the most appropriate value for the graph's parameter.

V. Conclusion

In this paper, we proposed a novel graph-based SSL algorithm, called RMGTHOR, by generalizing the RMGT algorithm for any p.s.d. matrix. Consequently, our method can naturally deal with higher order regularization, which is not the case of the original RMGT algorithm. We proved that our method admits a closed-form solution. We also proved that the RMGT algorithm is a special case of our method for the combinatorial Laplacian.

Our method showed good to exceptional performance in comparison to state-of-the-art methods for SSL. The results also showed that our method achieved good stability with respect to k in most data sets, which is a valuable property for real applications. However, the Laplacian's degree may have a moderate influence in the performance of our method. In the absence of other criteria for parameter selection, we suggest $p = 2$ as initial choice for real applications.

Acknowledgments

This research was supported by the Brazilian agencies CAPES, FAPESP under grant 2012/50714-7, and CNPq under grant 446330/2014-0.

References

[1] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*. The MIT Press, 2006.

[2] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *JMLR*, vol. 7, pp. 2399–2434, 2006.

[3] C. A. R. de Sousa, V. M. A. Souza, and G. E. A. P. A. Batista, "An experimental analysis on time series transductive classification on graphs," in *IJCNN*, 2015.

[4] W. Liu and S.-F. Chang, "Robust multi-class transductive learning with graphs," in *CVPR*, 2009, pp. 381–388.

[5] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, 2004, pp. 321–328.

[6] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.

[7] C. A. R. de Sousa, V. M. A. Souza, and G. E. A. P. A. Batista, "Time series transductive classification on imbalanced data sets: an experimental study," in *ICPR*, 2014, pp. 3780–3785.

[8] C. A. R. de Sousa, S. O. Rezende, and G. E. A. P. A. Batista, "Influence of graph construction on semi-supervised learning," in *ECML/PKDD*, 2013, pp. 160–175.

[9] R. Johnson and T. Zhang, "On the effectiveness of laplacian normalization for graph semi-supervised learning," *JMLR*, vol. 8, pp. 1489–1517, 2007.

[10] X. Zhou and M. Belkin, "Semi-supervised learning by higher order regularization," in *AISTATS*, 2011, pp. 892–900.

[11] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning," in *ICML*, 2009, pp. 441–448.

[12] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *JMLR*, vol. 12, pp. 1149–1184, 2011.

[13] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[14] M. Hein, J.-Y. Audibert, and U. von Luxburg, "Graph laplacians and their convergence on random neighborhood graphs," *JMLR*, vol. 8, pp. 1325–1368, 2007.

[15] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.

[16] C. A. R. de Sousa, "An overview on the gaussian fields and harmonic functions method for semi-supervised learning," in *IJCNN*, 2015.

[17] ——, "Impacto da geração de grafos na classificação semissupervisionada," Master's thesis, 2013.

[18] M. Hein and M. Maier, "Manifold denoising," in *NIPS*, 2007, pp. 561–568.

[19] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[20] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, pp. 1–30, 2006.